バス結合共有メモリ型マルチプロセッサにおけるバリア同期機構の評価

茶屋道 宏貴† 岩根 雅彦†

同期オーバヘッドを削減するためのMCAMを用いた重複を許さない任意参加型バリア同期機構とその利用法を提案する．同期機構を搭載したバス結合共有メモリ型マルチプロセッサMDBM/FMMを用いて，線状バリア，面状バリア，および線状バリアと面状バリアの組合せである面線バリア同期の評価を行った．doacrossの評価では，線状バリア同期に対し面状バリア同期の速度向上比は平均14.3%向上し，動作解析によってバリア領域とバストラフィックの関係を明確にした．一般的な応用プログラムの評価では，線状バリアに対し面線バリア同期の速度向上比は平均6.3%向上し，面線バリア同期では同期待ちが削減されることが判明した．また複数の並列化プログラムを多重実行させ，平均ターンアラウンド時間を測定した．複数の並列化プログラムを逐次実行した場合に対し，後続する並列化プログラムをアイドルプロセッサに埋め込み実行した場合，平均ターンアラウンド時間比が最良で13.4%向上し，最悪で46.6%低下した．同時にプロセッサへの並列化プログラムの割当てと平均ターンアラウンド時間の関係を明確にした．

Evaluation of Barrier Synchronization Mechanism on a Bus-based Multiprocessor

Hirotaka Chayamichi† and Masahiko Iwane†

The barrier synchronization mechanism using the modified CAM reduces the synchronization overhead. This barrier mechanism can perform the basic barrier, the range barrier, and the semi-range barrier. These barriers are evaluated on the bus-based multiprocessor MDBM/FMM. The experiment using the doacross loop shows that the average speedup ratio of the range barrier to the basic barrier is 1.14, also clarifies the relation between the barrier region and the bus waiting. The experiment using some general application programs shows that the average speedup ratio of the semi-range barrier to the basic barrier is 1.06 and the semi-range barrier can reduce the waiting time for the synchronization to the basic barrier. When some parallelized programs are performed simultaneously on MDBM/FMM, the case that the successor programs are assigned to the idle processors and the case that these programs are executed sequentially are evaluated. The results show that the former achieves the average turn-around ratio up to 13.4% to the latter, also clarifies the relation between the task assignment and the average turn-around time.

1. はじめに

バス結合共有メモリ型マルチプロセッサで協調して並列処理を行う場合，プロセッサ間の同期と通信のオーバヘッドが問題となる．同期に関する解決策の1つとして，プロセッサ間同期をハードウェア化したバリア同期があり，その拡張としてFuzzy Barrier[1]，Elastic Barrier[2]，Ultimate Barrier[3]，SBM[4]，DBM[5]，MSBM[6]，RBCQ[7]などがある．

Fuzzy Barrier[1]は領域のあるバリアを実現し，同期識別タグとマスクレジスタによって多重バリア発行を可能にしている．Elastic Barrier[2]では同期情報を格納するカウンタによってバリア同期による無駄な待ちを回避する．Ultimate Barrier[3]では2つの同期用カウンタによりバリア同期を重複可能にしている．SBM[4]では完全順序関係をバリアキューに登録することにより任意のプロセッサで同期をとる．DBM[5]では半順序関係をCAM（Content Addressable Memory）に登録する任意参加型同期である．RBCQ[7]はバリアキュー構造によるブロッキングを少なくした同期機構である．バリア同期モデルを分類定式化し，シミュレーションにより性能の定量的な比較考察がなされている[8]．

† 九州工業大学工学部電気工学科
Department of Electrical Engineering, Faculty of Engineering, Kyushu Institute of Technology

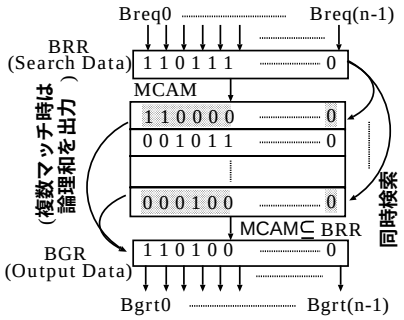

図 1 MCAM を用いた同期機構
Fig. 1 Synchronization mechanism using MCAM.

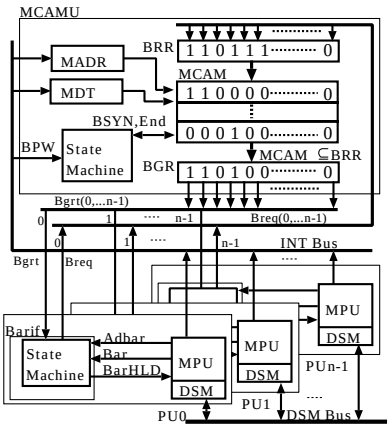

図 2 バリア同期機構を持った分散共有メモリ型マルチプロセッサ
Fig. 2 Distributed shard memory based multiprocessor on the barrier synchronization mechanism.

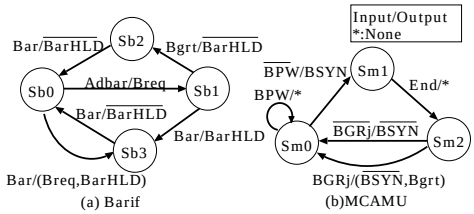

図 3 状態遷移図
Fig. 3 State diagram.

今回，MSBM[6]での重複を許さない任意参加型バリア同期機構をビット拡張し，同期機構の利用方法の検討を行い，マルチプロセッサ MDBM/FMM[9),10)]を用いて，同期機構の評価を行った．

本論文では MCAM（Modified CAM）を用いたバリア同期機構，同期機構の利用方法，同期機構での面状バリアと面線バリアについて述べる．続いて，基礎実験で同期と粒度の関係を示し，面状バリア，面線バリアの性能評価とともに，バストラフィックが性能に与える影響を解析した．さらにマルチタスクで実行したときのプロセッサへのタスク割当てと平均ターンアラウンド時間の評価および考察について述べる．

2. バリア同期機構

2.1 MCAM を用いた同期機構[6)]

MCAM を用いた同期機構はバリア同期要求レジスタ BRR，バリア同期完了レジスタ BGR および n ビット × n/2 の MCAM を持ち，その概要を**図 1** に示す．n はシステムのプロセッサ総数を表す．MCAM には MCAM データとして，同期をとるプロセッサ PU に対応したビットパターン（バリアパターン）を設定する．

BRR と BGR の各ビットはそれぞれ PU のバリア要求信号 Breq およびバリア許可信号 Bgrt に接続される．PU からの Breq 信号により BRR 内の該当ビットをセットする．BRR を検索データとして，MCAM 内に設定されたバリアパターンの全エントリを同時に検索する．式 (1) の包含関係が成立すれば該当バリアパターンが MCAM から BGR に読み出される．

$$\text{if } MCAM_{ij} \subseteq BRR_j \text{ then } BGR_j := MCAM_{ij} \quad (j = 0, \cdots, n-1) \tag{1}$$

ただし，i は MCAM 内 i 番目のエントリ，j は MCAM，BRR，BGR の j 番目のビットを表す．また包含関係は $(MCAM_{ij} \subseteq BRR_j) \equiv (\overline{MCAM_{ij}} \vee BRR_j)$ である．包含されるバリアパターンが MCAM 内に複数存在する場合，包含されるバリアパターンの論理和を BGR に出力する．BGR に格納されたデータが PU への同期成立信号となり，同時に BGR にセットされたビットと同じ BRR のビットをリセットする．

2.2 マルチプロセッサへの搭載

MCAM 同期機構 MCAMU を実装した分散共有メモリ型マルチプロセッサを**図 2** に示す．同期機構を持ったマルチプロセッサは MCAMU と PU からなる．MCAMU は MCAM のほかに MCAM アドレスレジスタ MADR と MCAM データレジスタ MDT を持つ．PU はバリア同期インタフェース Barif を持つ．

線状および面状バリア同期[11)]を実現するためプロセッサ側には Adbar 命令と Bar 命令を用意する．Adbar 命令と Bar 命令でバリア領域の最初と最後を表すことで面状バリア同期を実現し，Bar 命令単体では線状バリア同期を実現する．**図 3** に Barif と MCAMU の状態遷移図を示す．図 3 中の斜線 (/) の左側は入力で，右側は出力である．括弧は 2 出力を意味する．PU が Adbar 命令を実行すると Barif に対して Adbar 信号が発行される．Barif は図 3 (a) に示す状態遷移図に従って Sb0（PU は非バリア領域の命令を実行，初

期状態）から Sb1（PU はバリア領域の命令を実行，同期未完了）に遷移する．Sb1 で同期が成立すると，Bgrt 信号が送られ Sb2（PU はバリア領域の命令を実行，同期完了）に遷移する．Sb2 では PU が Bar 命令を実行し，Barif に対して Bar 信号が送られてくると Sb0 に遷移する．Sb1 で同期成立前に PU が Bar 命令を実行すると，Barif に対して Bar 信号が送られ Sb3（PU はバリア領域の命令を実行完了，同期の完了待ち）に遷移し，同期成立後 Sb0 に遷移する．PU が Bar 命令を実行すると Barif に対して Bar 信号が送られ，Barif は Sb0 から Sb3 に遷移する．Sb3 では，PU は実行を中断し，Bgrt 信号が送られてくると実行を再開する．

バリアパターン更新（設定/解放）のためにプロセッサ側に MCAMW 命令を用意する．PU が MCAMW 命令を実行すると，MCAM 内アドレス，バリアパターンおよびバリアパターン書き込み信号 BPW を INT Bus を通じて MCAMU に送る．MCAMU は図 3 (b) に示す状態遷移図に従って Sm0（初期状態）のとき BPW=1 ならばバリアパターン書き込みが終了するまで Sm0 を繰り返す．Sm0 のとき BPW=0 なら Sm1（同期成立検出）へ遷移する．Sm1 ではバリアパターン検索終了を表す End 信号が有効になると Sm2（同期成立検出完了）へ遷移する．Sm2 では検索結果に応じて Bgrt 信号を PU に送り，Sm0 へ遷移する．なお，Barif と MCAMU は独立に動作する．

3. バリアグループにおける面線バリア同期

3.1 バリアグループの構成

バリア同期をとる PU の組をバリアグループと呼び，バリアグループに属する PU をバリアパターンとして MCAM に設定する．バリアグループの概念図を**図 4** に示す．1 点破線が並列化プログラム，×印がバリアパターンの設定，網掛線が設定されたバリアパターン，横線がバリア，および点線がダミーバリアを示す．図 4 に示すように必要に応じた並列度でバリアグループを設定し，並列化プログラムの実行中にバリアパターンの変更を行う．ただし，同じ PU は同時に異なるバリアパターンに存在することはできない．先行バリアグループを BG_i，後続バリアグループを BG_{i+1} とすればバリアパターン設定プロセッサ PU_S は $PU_S = BG_i \cap BG_{i+1}$ である．ただし，$BG_i = \phi$ ならば $BG_i = U$（全集合）である．なお今後の議論では並列化プログラム実行中はコンテキストスイッチングが起こらないものとする．

並列化プログラムの実行において，バリアグループ

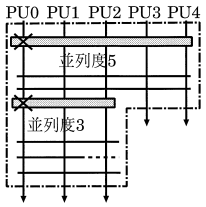

図 4 バリアグループ
Fig. 4 Barrier group.

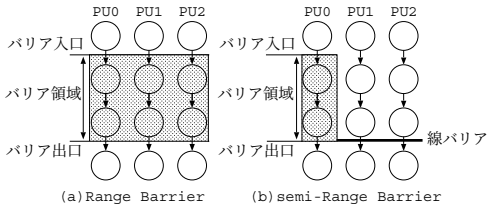

図 5 面状バリアと面線バリア
Fig. 5 Range barrier and semi-range barrier.

BG_i でのバリア同期間の平均実行時間を T_{e_i}，バリア同期時間を T_b，バリアパターン更新時間を T_{bg}，BG_i でのバリア同期回数を n_i，実行するバリアパターンの数を m とすると，並列化プログラムの実行時間 T は次式となる．

$$T = \Sigma_{i=1}^{m}(T_{bg} + (n_i + 1) \cdot T_{e_i} + n_i \cdot T_b) \quad (2)$$

また p 台で並列化した複数のプログラムを同時に投入してからの各並列化プログラムの終了時間を T_{t_i}，並列化プログラム数を M とすると，平均ターンアラウンド時間 T_{avg} は次式となる．

$$T_{avg} = \Sigma_{i=1}^{M} T_{t_i}/M \quad (3)$$

複数のプログラムを 1 台のプロセッサで連続実行したときの平均ターンアラウンド時間を T_{Savg}，複数のプログラムをそれぞれたかだか p 台で並列化し，順次実行したときの平均ターンアラウンド時間を T_{Pavg} とすれば平均ターンアラウンド時間比 T_{ratio} は次式となる．

$$T_{ratio} = T_{Pavg}/T_{Savg} \quad (4)$$

3.2 面線バリア

バリアグループによるバリア同期機構の利用方法に，線状バリア，面状バリア，面線バリアがあり，面状バリアと面線バリアの概念図を**図 5** に示す．線状バリアではすべての同期点において線バリアのみで同期をとる．したがって，同期に参加する全 PU がバリアに到達すると同期が成立し，PU_i がバリアに到達した時点で同期が不成立なら，アイドル状態となる．面状バリアではすべての同期点においてバリア領域を持つ面バリアによって同期をとる．PU_i がバリア領域の入

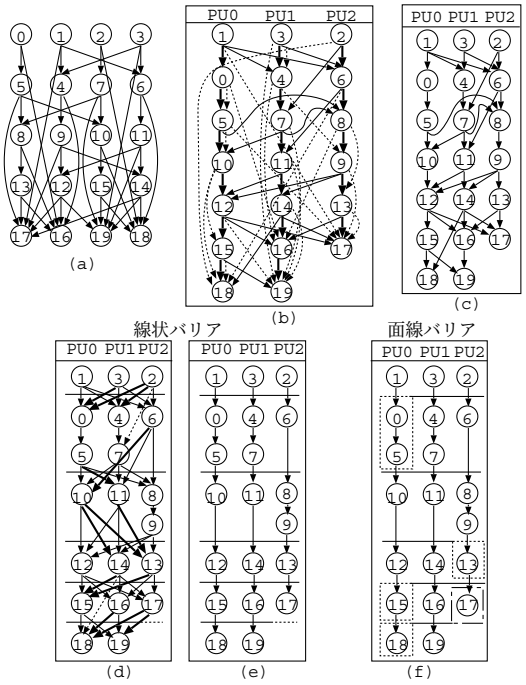

図 6 線状バリアと面線バリア
Fig. 6 Basic barrier and semi-range barrier.

口に到達した時点で同期が不成立でもアイドル状態にならない．PU_i がバリア領域の出口に到達した時点で同期が不成立ならアイドル状態となり同期成立を待つ[8]．面線バリアは図 5 (b) に示すように，バリア領域を持つ面バリアと通常の線バリアの組合せである．

図 6 (a) に示すようなタスクグラフを 3PU で並列実行するため，PU_i (i=0,1,2) に対して図 6 (b) のようにタスク割当てを行った．円はタスク，数字はタスク番号，矢印は先行制約を表す．矢印の始点を先行制約元タスク T_{src}，矢印の終点を先行制約先タスク T_{dst} とする．

タスクをプロセッサに割り当てたことによりタスクに実行順序が生じ，図 6 (b) に示す太矢印の先行制約が生じる．また，図 6 (b) でタスク 1 と 6 の間およびタスク 1 と 17 の間に先行制約があり，タスク 6 と 17 の間にもタスク割当てによって生じた先行制約がある．したがって，タスク 1 と 6 の間およびタスク 6 と 17 の間の先行制約を満たせばタスク 1 と 17 の間の先行制約は保証され，その先行制約を省略できる．同様に，タスクの実行順序により保証される先行制約も省略できる[7]．図 6 (b) の点線矢印が省略できる先行制約であり，それらの先行制約を省略した結果は図 6 (c) のようになる．

原始的な線状バリアでは図 6 (c) をもとに，T_{src} の実行後および T_{dst} の実行前にバリアを設ける．バリアを設けたことによって図 6 (d) での太矢印の先行制約が生成される．横線は線バリア，横点線はダミー線バリアを意味する．新たな先行制約が追加されたことにより，先行制約が保証されるものは削除し，先行制約が省略できなくなるまで繰り返す．その最終結果を図 6 (e) に示す．

面線バリアでは，図 6 (e) の線状バリアをもとに，先行制約を見直し発見的にバリア領域を設けた．その最終結果を図 6 (f) に示す．点線枠が面バリア，一点破線枠がダミー面バリアである．

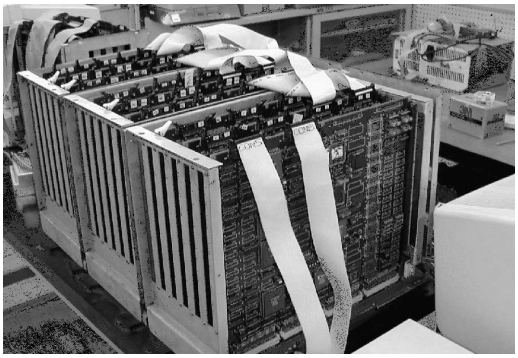

図 7 MDBM/FMM 写真
Fig. 7 Picture of MDBM/FMM.

4. 基礎実験

4.1 実験環境

MCAM を用いたバリア同期機構の評価を行うために図 2 を具現化したマルチプロセッサ MDBM/FMM[9),10)]を開発した．MDBM/FMM の写真を**図 7** に示す．MDBM/FMM は集中型バリア同期機構と n 台 (n=32) の PU からなる．MCAM のハードウェア量は $n^2/2$=512 ビット，MCAM を含めた同期機構全体のハードウェア量は約 13,000 ゲートである．また PU は DPM（Dual Port Memory）で構成された分散共有メモリ DSM（Distributed Shard Memory）を持つ．

バリア同期命令の実行時間（クロック数）は図 3 に示す Barif と MCAMU の状態遷移過程で異なる．その最大値 T_{bmax} は，Barif が Sb0, Sb3, MCAMU が Sm1, Sm2, Sm0, Sm1, Sm2 と遷移したときの値であり，最小値 T_{bmin} は，Barif が Sb0, Sb3, MCAMU が Sm0, Sm1, Sm2 と遷移したときの値である．MCAM の検索は Sm1 で行われるが，バリアパターン数，PU 数に関係なく検索時間は一定である．バリア同期命令の設計値 T_{bn} を T_{bmax} と T_{bmin} の平均とし，以下の

表 1 評価用マシン基本データ
Table 1 Basic performance of MDBM/FMM.

種別	設計値	種別	設計値
Bar	1.000	ADD	1.478
Adbar	0.913	SUB	1.478
MCAMW(バリアパターン書換)	12.478	MUL	6.804
DSM Read Bus Cycle(1 byte)	0.218	DIV	8.000
DSM Read Bus Cycle(2 byte)	0.435	FADD	7.348
DSM Read Bus Cycle(4 byte)	0.870	FSUB	7.348
DSM Write Bus Cycle(1 byte)	0.466	FMUL	9.565
DSM Write Bus Cycle(2 byte)	0.932	FDIV	12.348
DSM Write Bus Cycle(4 byte)	1.864		

Serial

```
Register int R1 = 0;
sum=0,R1=0;
for(i=0; i<n; i++){
    R1=R1+a(i);
}
sum=R1;
```

(a)

Parallel

```
PID=0,1,2,…,n-1
 register int R1 = 0;
 if(PID=0)sum=0;
 R1=0;
 for(i=PID; i< n/P (PID+1) ; i++){
    R1=R1+a(i);
 }
 if(PID%2=1)S(PID)=R1;                    // (1)
 Bar;
 for(i=1; i<=⌈ log2P ⌉ i++){
    if(PID%2^i=0)R1=R1+S(PID+2^(i-1)); // (2)
    if((PID/2^i)%2=1)S(PID)=R1;        // (3)
    Bar;
 }
 if(PID=0)sum=R1;
```

(b)

図 8 配列要素の和計算プログラム
Fig. 8 Summation program.

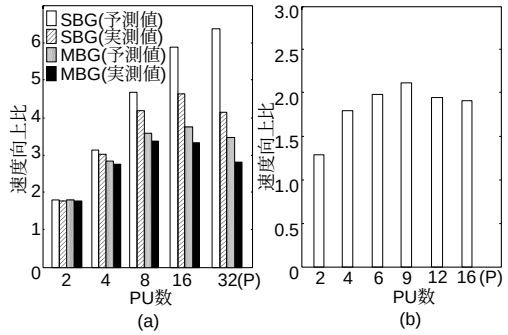

図 9 配列要素の和と多項式の速度向上比
Fig. 9 Speedup ratio of summation program and polynomial expression.

表 2 各要素の実行時間
Table 2 Execution time of each element.

		SBG			MBG		
i	T_p	T_{bt}	T_{nd}	T_{bgt}	T_{bt}	T_{nd}	T_{bgt}
2	198.4	1.0	6.4	12.7	1.0	6.4	12.7
4	101.8	2.0	6.4	12.7	2.0	6.4	25.5
8	54.0	3.0	7.6	12.7	3.0	7.6	38.2
16	30.6	4.0	8.2	12.7	4.0	8.2	51.0
32	19.5	5.0	9.1	12.7	5.0	9.1	63.7

議論では実行時間の値を T_{bn} を 1 として正規化する．バリア同期命令の実測値は平均で 1.000 となった．評価に用いた MDBM/FMM の基本データを**表 1** に示す．左側は Bar 命令，Adbar 命令，MCAMW 命令および共有メモリの読み出し/書き込みのバスサイクル，右側は整数/浮動小数点の四則演算である．

評価プログラムの実行にあたって，最初にバリアパターンを設定し実行中はバリアパターンを変更しない方法 SBG と実行中に適宜バリアパターンを変更する方法 MBG で実験を行った．

4.2 粒度と同期

配列要素の和を求める**図 8** (a) のプログラムを，図 8 (b) のように，各部分和を計算し，その総和を 2 分木計算として求める形で並列化した．n を配列要素数，配列 $a(i)$ のデータ型は整数型，PID はプロセッサ ID，P はプロセッサ使用台数，S は共有変数とする．逐次版のループ 1 回の実行時間を T_{sl}，バリア同期命令実行時間を T_b，2 分木の各ノードの平均実行時間を T_{nd}，バリアパターン更新時間を T_{bg}，バリアパターン更新回数を m とする．実行時間の予測値 T_{sum} は次式となる．

$$T_{sum} = \lceil n/P \rceil \cdot T_{sl} + \lceil \log_2 P \rceil (T_{nd} + T_b) + m \cdot T_{bg} \quad (5)$$

n=64 のときの逐次版に対する並列版の予測値と実測値の速度向上比を**図 9** (a) に示す．式 (5) の予測値の算出では $T_{sl} = 6.1$，$T_{nd} = 6.4$ である．SBG では m=1，MBG では各ノード列をバリアグループとし，$m = \lceil \log_2 P \rceil$ とした．また，並列度 i の部分和計算の実行時間 T_p，全同期時間（同期数）T_{bt}，T_{nd}，全バリアパターン更新時間 T_{bgt} の実測値を**表 2** に示す．

並列度 16 における平均粒度は 11.4 となり，そのときの SBG における速度向上比は実測値で 4.6 となった．

部分和計算では，$P/2$ 台の PU が各々が計算した部分和をいっせいに共有メモリへ書き込むため（図 8 の (1)），バス書き込み（DSM Write Bus Cycle）がシリアライズされた．1 回のバス書き込み時間を T_{wait} とすると，部分和の実行時間は次式のように考えることができる．

$$T_p = \lceil n/P \rceil \cdot T_{sl} + P/2 \cdot T_{wait} \quad (6)$$

PU に対して $(P/2-1) \cdot T_{wait}$ がバス書き込み時の最大バス待ち時間である．T_{nd} は 2 分木の各ノードの平均実行時間であり，予測値では並列度に関係なく一定であるが，実測値では表 2 から並列度の増加とともに同じか大きくなっている．これは図 8 (b) の (2), (3) の実行時に共有メモリへのアクセスが逐次化されたことによる．

MBG の場合は，各ノード列ごとにバリアパターン

の書き換えを行った．表 2 に示すような T_{nd} に対し，バリアパターン書き換えを行うと，書き換え自体がオーバヘッドとなる．したがって，T_{bgt} の値が全体の実行時間に対して無視できるようにバリアグループを形成する必要がある．

次に sin 関数を Taylor 展開した多項式 (7) を，P 台（P=2〜16）の PU で計算時間が均等になるように分割する．

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{7}$$

式 (7) を p 個に分割した部分式は次のようになる．

$$f_i = (a_{m_{i+1}-1} \cdot x^{m_{i+1}-m_i-1} + \cdots + a_{m_i})x^{m_i} \tag{8}$$

ただし，m_i を部分式の項数，$i = 0, 1, \cdots, p-1$，$m_0 = 0$ とする．各部分式の括弧内はホーナー法で計算する．k 次多項式において，乗算時間を T_a，加算を T_b とすると，式 (8) の括弧部分の実行時間は $k \cdot (T_a + T_b)$ となる．また x^{m_i} の計算時間は $T_a \cdot (2 \log_2 m_i)$ となり[13]，各部分式の計算時間 T_{m_i} は次式となる．

$$T_{m_i} = k \cdot (T_a + T_b) + T_a \cdot (2 \log_2 m_i) \tag{9}$$

式 (9) より，$\Sigma_{i=1}^{p-1} \mid T_{m_{i+1}} - T_{m_i} \mid$ が最小となる m_i で分割する．各部分式の計算後に同期をとり，部分式の和を 2 分木の計算として求める．

並列化した多項式による実験結果を図 9 (b) に示す．並列度 9 のとき最も速度向上比が良く，速度向上比は 2.1 となった．配列要素の和と同様にバス競合のため台数効果が得られなかった．

4.3 面状バリア

図 10 (a) のプログラムにコード入れ換え[1]を適用し，図 10 (b) では線状バリア，図 10 (c) では面状バリアにより doacross として並列化した[12]．ただし，$a(i)$，$b(i)$，$c(i)$ のデータ型は整数型，PID はプロセッサ ID，$a(i)$ は共有変数とする．逐次版のループ 1 回の実行時間を T_{sl}，バリア同期命令実行時間を T_b，Adbar 命令実行時間を T_{ad}，およびバリアパターン更新時間を T_{bg} とする．線状バリアでの予測値の実行時間 T_B は次式となる．

$$T_B = \lceil n/P \rceil \cdot (T_{sl} + T_b) + T_{bg} \tag{10}$$

面状バリアでの予測値の実行時間 T_R は次式となる．

$$T_R = \lceil n/P \rceil \cdot (T_{sl} + T_{ad} + T_b) + T_{bg} \tag{11}$$

n=8192 のときの逐次版に対する予測値と実測値の速度向上比を**図 11** に示す．式 (10)，(11) の予測値の算出では $T_{sl} = 8.4$ である．

並列度 4 と並列度 8 での面状バリアでの doacross の動作解析結果を**図 12** に示す．図 12 から，並列度 4 では，バス待ちのために PU0 に対して，PU1，PU2，

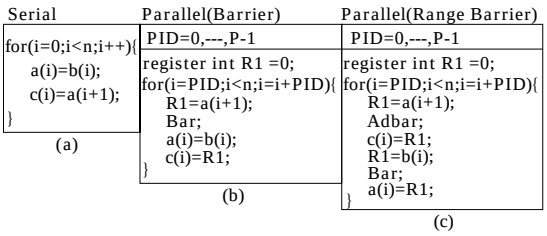

図 10 doacross プログラム
Fig. 10 Doacross program.

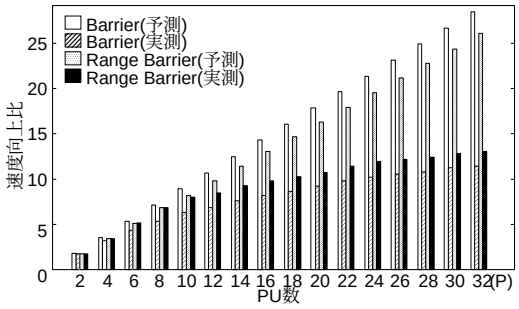

図 11 doacross プログラムの速度向上比
Fig. 11 Speedup ratio of doacross program.

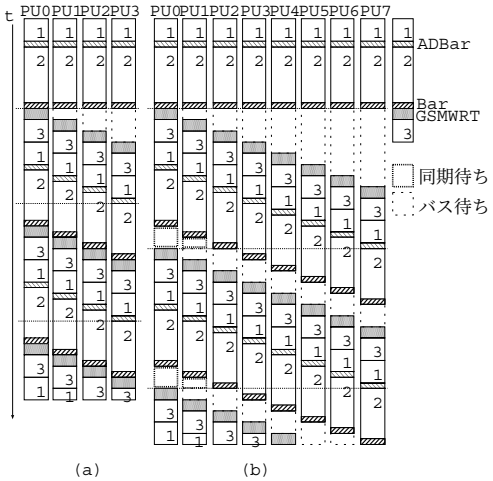

図 12 面状バリアの動作解析
Fig. 12 Analysis of range barrier.

P3 の実行が遅れている．しかし，全 PU は Adbar 命令をバリア領域内で実行しており同期待ちが生じず，最初のバス競合以降でバス競合は発生しない．一方，並列度 8 では，PU0 に対して PU7 の実行がバス待ちにより大きく遅れ，PU7 はバリア領域外で Adbar 命令を実行している．そのため，同期待ちが生じ，それ以降においてもバス競合が発生している．並列度 4 では，実行のずれがバリア領域より小さく，同期待ちは生じていない．一方，並列度 8 では実行のずれがバリア領域より大きく，同期待ちが生じた．バリア領域時

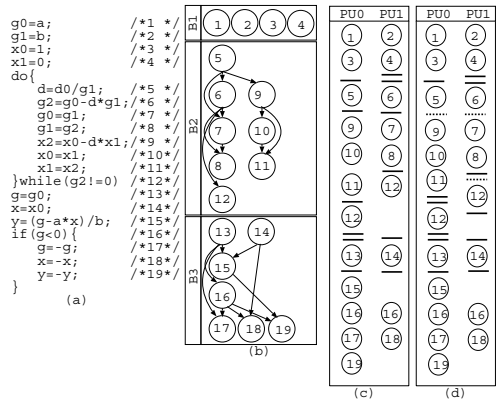

図 13 最大公約数
Fig. 13 Greatest common divisor.

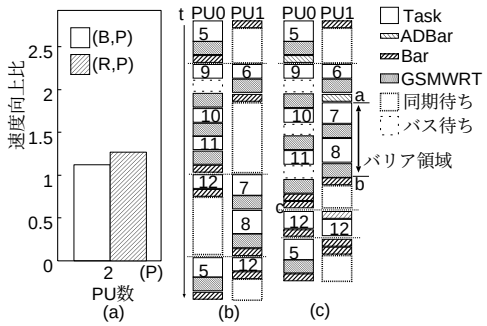

図 14 最大公約数の実験結果
Fig. 14 Results of greatest common divisor.

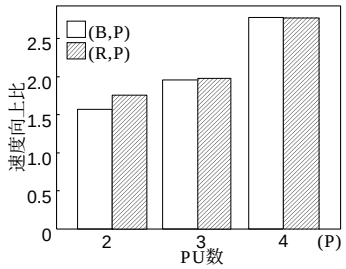

図 15 微分方程式解法の速度向上比
Fig. 15 Speedup ratio of differential equation.

間（Adbar 命令から Bar 命令までの時間）を T_r とすると，バス競合時に式 (12) を満たせば同期待ちが起こらない．

$$T_r > (P-1) \cdot T_{wait} \tag{12}$$

したがって，式 (12) を満たす限り，式 (11) の予測値と実測値がほぼ近くなるが，式 (12) を満たさなければ，同期待ちにより予測値と実測値の差が大きくなる．実験結果では $T_r = 3.012$，$T_{wait} = 0.466$ となった．

線状バリアと比較すると，面状バリアの速度向上比は平均 14.3%向上した．面状バリアではバリア領域を設けることで，同期点に広がりができ，バス待ちの影響が緩和したためである．

5. 単一プログラムによる実験

5.1 面線バリア

図 13 (a) の最大公約数のプログラムを，文 1〜4 をブロック B1，文 5〜12 を B2，文 13〜19 を B3 とし，文をノードに持つ図 13 (b) のタスクグラフをもとに 2 台で並列化し，SBG として実行する．線状バリアと面線バリアでの Adbar/Bar 命令の挿入結果を図 13 (c), (d) に示す．横点線が Adbar 命令，横実線が Bar 命令を示す．

最大公約数の逐次版に対する線状バリアと面線バリアの速度向上比を**図 14** (a) に示す．速度向上比は線状バリアでは 1.12，面線バリアでは 1.27 となった．並列度 2 での B2 の動作解析結果を線状バリアは図 14 (b)，面線バリアは図 14 (c) に示す．線状バリアでは図 14 (b) の PU1 のタスク 6 と PU0 のタスク 12 の実行後に同期待ちが生じているが，面線バリアではバリア領域によってそれらの同期待ちが削減されている．したがって，線状バリアに対し面線バリアの速度向上比が 13.0% 向上した．また，図 14 (c) においてタスク 8 の実行後に同期待ちが生じている．これは，a の Adbar 命令に対する Bar 命令が b，c であり，a と b 間のバリア領域内で c が実行されていないためである．

次のような 2 元連立 2 階微分方程式を考え，4 次のルンゲクッタ法[14]での解法を考える．

$$d^2x_1/dt^2 = -2x_1 + x_2 \tag{13}$$
$$d^2x_2/dt^2 = x_1 - 2x_2 \tag{14}$$

線状バリアおよび面線バリアについて，文レベルで P 台（P=2,3,4）で並列化し，SBG として実行する．

微分方程式解法の逐次版に対する速度向上比を**図 15** に示す．線状バリアと比較して，面線バリアは平均 4.1% 向上した．並列度 2 は最大公約数と同様バリア領域を設けることで線バリアよりも面線バリアの速度向上比が向上した．並列度 3，4 は有効なバリア領域ができず線状バリアと面線バリアの速度向上比はほぼ等しい．

5.2 応用プログラムでの評価

5.2.1 実　　験

次のような 2 元連立 2 階微分方程式を考え，4 次のルンゲクッタ法での解法を考える．

$$d^2x_1/dt^2 = -2x_1 + x_2 \tag{15}$$
$$d^2x_2/dt^2 = x_1 - x_2 + F\sin(\omega t) \tag{16}$$

sin 関数の値はあらかじめ表として求めておく．

表を作成するときの sin 関数は，4.2 節と同様にし

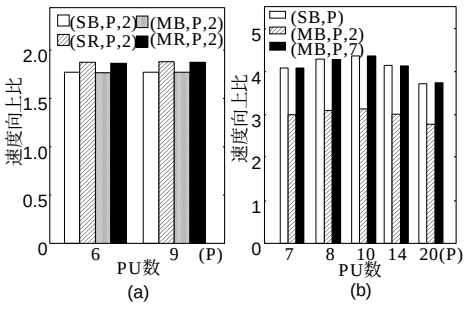

図 16 微分方程式解法と判別分析の速度向上比
Fig. 16 Speedup ratio of differential equation and discriminant analysis.

て 6，9 台で並列化した．ルンゲクッタ法の計算では線状バリア，面線バリアについて，文レベルで 2 台で並列化を行った．SBG では sin 関数計算とルンゲクッタ法を BG_1 に割り当て，MBG では sin 関数計算を BG_1 に，ルンゲクッタ法を BG_2 に割り当てる．

逐次版に対する速度向上比を**図 16** (a) に示す．図 16 の括弧内の第 1 要素はバリアグループ形態 S (SBG)/M (MBG) とバリアの種類 B (線状バリア)/R (面線バリア) である．第 2 要素は BG_1 の PU 台数，第 3 要素は BG_2 の PU 台数である．

次に判別される群数を q，各群の標本数を n，標本変数の数を r とする．n 個の標本を構成する変数データから群 G_k の変数の平均 $\bar{x}_m^{(k)}$（式 (17)），群 G_k の変数の分散，共分散 $s_{ml}^{(k)}$（式 (18)）を求め，r 変数を得る．ただし，$q = 2$，$n = 40$，$r = 7$ とする．次に r 変数より群 G_k の変数の共通の分散，共分散 s_{ij}（式 (19)），相関行列 r_{ij}（式 (20)），相関行列の逆行列 V^{-1}，群 G_k の変数のマハラビノス平方距離の和 $D_{(k)}$（式 (21)）を求め，判別関数 z（式 (22)）により判別する[15]．

$$\bar{x}_m^{(k)} = 1/n \cdot \Sigma_{i=1}^{n} x_{mi}^{(k)} \quad (k = 1, 2, m = 1, \cdots, 7) \tag{17}$$

$$s_{ml}^{(k)} = \Sigma_{i=1}^{n}(x_{mi}^{(k)} - \bar{x}_m^{(k)})(x_{li}^{(k)} - \bar{x}_l^{(k)})/n \quad (k = 1, 2; m, l = 1, \cdots, 7) \tag{18}$$

$$s_{ij} = \frac{(n-1)s_{ij}^{(1)} + (n-1)s_{ij}^{(2)}}{n+n-2} \quad (i, j = 1, \cdots, 7) \tag{19}$$

$$r_{ij} = s_{ij}/(\sqrt{s_{ii}}\sqrt{s_{jj}}) \quad (i, j = 1, \cdots, 7) \tag{20}$$

$$D_{(k)} = \mathbf{a}^{(k)} V_{(k)}^{-1} ({}^t\mathbf{a}^{(k)}) \quad (k = 1, 2) \tag{21}$$

$$z = D_{(1)}^2 - D_{(2)}^2 \tag{22}$$

ただし，$\mathbf{a}^{(k)} = (a_1^{(k)}, a_2^{(k)}, \cdots, a_7^{(k)})$，$a_i^{(k)} = (x_b - \bar{x}_m^{(k)})/s_{ml}^{(k)}$ である．

式 (17)，(18) を Doall で並列化し，群ごとの集計部分を 2 台で並列化した．式 (19)〜(21) をそれぞれ Doall として 2，7 台で並列化した．式 (19)〜(21) の各式の計算ごとに同期をとり，最後に各 PU で計算した値をもとに 1 台で判別する．SBG では式 (17)〜(22) を BG_1 に割り当て，式 (19)〜(22) の並列度を 7 とした．DBG では式 (17)，(18) を BG_1 に，式 (19)〜(22) を BG_2 に割り当てる．

逐次版に対する SBG と MBG の速度向上比を図 16 (b) に示す．括弧内の第 1 要素はバリアグループ形態とバリアの種類，第 2 要素は BG_1 の PU 台数，第 3 要素は BG_2 の PU 台数である．なお，判別分析では面線バリアは適用できなかった．

5.2.2 考　　察

微分方程式解法の実験結果では，SBG と比較して，MBG の速度向上比は平均 0.3%低下した．線状バリアと比較して面線バリアを用いた場合，速度向上比は平均 5.3%向上した．逐次実行と比較して SBG と MBG の速度向上比は向上した．(SB,P) では，sin 関数の計算を完了した PU にルンゲクッタ法の実行においてダミーバリアを挿入した．(MB,P,2) と (MR,P,2) ではバリアグループを変更することによって，sin 関数の計算完了後に (P-2) 台の PU がアイドル状態となった．(MB,P,2) と (MR,P,2) の各 P での BG_1 の平均実行時間は 15206，BG_2 の平均実行時間は 9793 となった．バリアパターン変更時間に対して，BG_1 や BG_2 での実行時間は大きい．sin 関数計算やルンゲクッタ法でのバリアパターン変更頻度であれば，バリアパターン変更によるオーバヘッドの影響は少なく，変更を行っても速度低下は小さい．

判別分析の実験結果では，(SB,P) と (MB,P,7) の速度向上比の差はほとんどなく，(MB,P,2) の速度向上比は (SB,P) に対して低い．これは BG_2 の最良並列度が 7 であることによる．また，速度向上比は (MB,10,7) が最も良い．BG_1 の並列化は基本的に Doall で行っており，最良並列度は 10 となる．(SB,P) ではアイドル PU にダミーバリアを挿入し，(MB,P,2) と (MB,P,7) では何もしない．(MB,P,7) と (MR,P,7) の各 P での BG_1 の平均実行時間は 25010，BG_2 の平均実行時間は 8671 となり，バリアパターン変更によるオーバヘッドの影響は少ない．

6. 並列化プログラムの多重実行による実験

6.1 実　　験

使用できるプロセッサは P 台とする．SBG では 5.2.1 項における sin 関数とルンゲクッタ法を BG_1

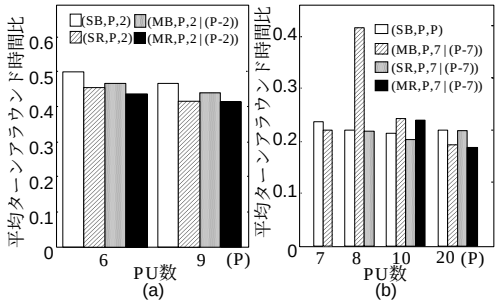

図 17 並列化プログラムの多重実行の実験結果
Fig. 17 Results of multiple parallelized programs execution.

に，doacross を BG_2 に割り当て，各プログラムを連続実行させる．MBG では sin 関数計算を BG_1 に，ルンゲクッタ法を BG_2 に，および doacross を BG_3 に割り当てる．まず，sin 関数を実行し，その実行中に生じたアイドル PU に doacross を実行させる．そして，sin 関数計算の実行後にルンゲクッタ法と doacross を同時に実行させる．ルンゲクッタ法に線状と面線バリア，doacross に線状と面状バリアを適用して実験を行った．SBG では BG_1 と BG_2 を P 台（P=6, 9），MBG では BG_1 を P 台，BG_2 を 2 台，および BG_3 を $P-2$ 台としたときの平均ターンアラウンド時間比を**図 17** (a) に示す．括弧内の第 1 要素はバリアグループ形態とバリアの種類，第 2 要素は BG_1 の PU 台数，第 3 要素は SBG では BG_2 の PU 台数，MBG では並列実行する BG_2 と BG_3 の PU 台数であり，縦線 (|) で区別した．

次に SBG では判別分析の式 (17)〜(22) を BG_1，doacross を BG_2 に割り当て，各プログラムを連続実行させる．MBG では判別分析の式 (17)，(18) を BG_1 に，式 (19)〜(22) を BG_2 に，doacross を BG_3 に割り当てる．判別分析の式 (17)，(18) の実行中に，アイドルプロセッサに doacross を実行させ，判別分析の式 (17)，(18) の実行後に式 (19)〜(22) と doacross を同時に実行させる．判別分析に線状バリア，doacross に線状と面状バリアを適用して実験を行った．SBG では BG_1 と BG_2 を P 台（P=7, 8, 10, 20），MBG では BG_1 を P 台，BG_2 を 7 台，BG_3 を $P-7$ 台とした場合の平均ターンアラウンド時間比を図 17 (b) に示す．図 17 (b) に示す括弧内の意味は微分方程式解法と doacross を用いた場合と同様である．

また，判別分析の実行中に生じるアイドルプロセッサに，配列要素の和，最大公約数および式 (13)，(14) の微分方程式解法を実行させる．各プログラムの PU への割当てと実行形態を**図 18** に示す．判別分析の

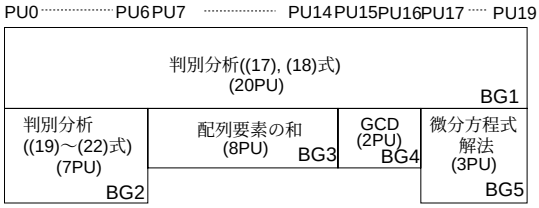

図 18 多重実行時の並列化プログラム割当て
Fig. 18 Assignment of multiple parallelized programs execution.

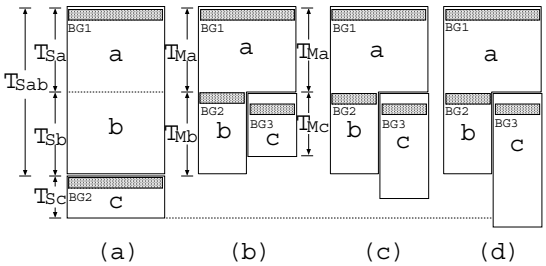

図 19 並列化プログラムの多重実行時の終了時間
Fig. 19 Final times of multiple parallelized programs execution.

式 (17)，(18) の計算を BG_1，式 (19)〜(22) の計算を BG_2 に割り当て，配列要素の和を BG_3，最大公約数を BG_4，微分方程式解法を BG_5 に割り当てる．判別分析の式 (17)，(18) の実行後に，残りのプログラムを同時に実行させる．各 BG の PU 台数は図 18 のとおりである．

6.2 考　　察

微分方程式解法と doacross を用いた実験で，MBG は SBG と比較して平均ターンアラウンド時間比が線状バリアで平均 6.5%，面線バリアで平均 1.9%向上した．

SBG の BG_1 での実行時間を T_{Sab}，BG_2 での実行時間を T_{Sc} とする．また MBG の BG_1 を T_{Ma}，BG_2 での実行時間を T_{Mb}，BG_3 での実行時間を T_{Mc} とする．SBG でのすべての終了時間 T_{t_S} と MBG でのすべての終了時間 T_{t_M} は次式となる．

$$T_{t_S} = T_{Sab} + T_{Sc} \quad (23)$$

$$T_{t_M} = T_{Ma} + max(T_{Mb}, T_{Mc}) \quad (24)$$

SBG での終了時間の概要を**図 19** (a)，MBG での p 終了時間の概要を図 19 (b)〜(c) に示す．実線がプログラム，網掛線が設定されたバリアパターンを表す．SBG の sin 関数計算の実行時間を T_{Sa}，SBG のルンゲクッタ法の実行時間を T_{Sb} とすれば $T_{Sab} = T_{Sa} + T_{Sb}$ である．

$T_{Mb} \geq T_{Mc}$ のとき T_{t_M} は次式となる（図 19 (b)）．

$$T_{t_M} = T_{Ma} + T_{Mb} \quad (25)$$

$T_{Sb} + T_{Sc} \geq T_{Mb}$ のとき $T_{t_S} \geq T_{t_M}$ (26)

$T_{Mb} < T_{Mc}$ のとき T_{t_M} は次式となる（図 19 (c), (d)）.

$$T_{t_M} = T_{Ma} + T_{Mc} \quad (27)$$

$T_{Sb} + T_{Sc} \geq T_{Mc}$ のとき $T_{t_S} \geq T_{t_M}$ (28)

$T_{Sb} + T_{Sc} < T_{Mc}$ のとき $T_{t_S} < T_{t_M}$ (29)

実験結果より微分方程式解法では SBG と MBG は同じ実行並列度であり，実行時間は $T_{Sab} \simeq T_{Ma} + T_{Mb}$ となる．(MB,6,2|4) と (MB,9,2|7) では式 (28) が成立し，(MR,6,2|4) と (MR,9,2|7) でも同様に式 (28) が成立した．

判別分析と doacross プログラムを用いた場合の SBG の BG_1 での実行時間を T_{Sab}，BG_2 での実行時間を T_{Sc} とする．また MBG の BG_1 での実行時間を T_{Ma}，BG_2 での実行時間を T_{Mb}，BG_3 での実行時間を T_{Mc} とする．実験結果より (MB,20,7|13) および (MR,20,7|13) は式 (28) が成立した．SBG と比較して平均ターンアラウンド時間比が 15.5%向上した．一方，(MB,8,7|1) と (MB,10,7|3) および (MR,8,7|1) と (MR,10,7|3) では式 (29) が成立したため，SBG と比較して平均ターンアラウンド時間比が平均 29.0%低下した．

線状バリア，面状バリア，面線バリアを用いたバリアグループを同時実行した結果，同期機構の独立性が確認できた．

また，図 18 の実行形態での各プログラムの実行時間に対して，各々のプログラムを単一で実行させたときの実行時間を比較した．その結果，図 18 における評価実行時間は，判別分析の BG_2 では 4.2%，配列要素の和では 9.3%，最大公約数では 6.3%，微分方程式解法では 3.5%増加した．プログラムを多重同時実行した場合，バストラフィックが原因で実行時間が平均 5.8%増加した．

7. むすび

バス結合共有メモリ型マルチプロセッサにおける MCAM を用いたバリア同期機構とその利用法を提案し，同期機構とバリア型同期の評価を行った．

実験では配列要素の和の並列化で MCAM によるバリア同期機構での同期と粒度の関係を示した．doacross の実験では，面状バリアは線状バリアに対してバストラフィックを削減でき，平均 14.5%の速度向上を得た．また動作解析して，バリア領域とバストラフィックの関係を明確にした．単一プログラムの実験では，面線バリアは線状バリアに対して同期待ちが削減でき，平均 6.3%の速度向上を得た．同時に動作解析した結果，面線バリア同期において同期待ちが削減されることが判明した．複数の異なる並列化プログラムを連続実行した場合とアイドルプロセッサを利用して実行した場合の実験を行った．MBG の平均ターンアラウンド時間比は SBG と比較して向上した場合と低下した場合があった．そして，実行時のプロセッサへのタスク割当てと平均ターンアラウンド時間の関係を明確にした．

今後の課題は面線バリア挿入アルゴリズムの検討，バリアグループの構成方法およびアイドルプロセッサを有効利用するためのスケジューリング方法の検討である．

謝辞 本研究を遂行するにあたり多大なご助力ならびご支援いただいた九州工業大学工学部山脇彰助手，（株）東芝北九州工場ならびに（株）春日製作所の方々に深謝いたします．

参考文献

1) Gupta, R.: The Fuzzy Barrier: A Mechanism for High Speed Synchronization of Processors, *Proc. 3rd Int. Conf. on ASPLOS*, pp.54–63 (Apr. 1989).
2) 松本 尚：Elastic Barrier：一般化されたバリア型同期機構，情報処理学会論文誌，Vol.32, No.7, pp.886–896 (1991).
3) 高木ほか：重複可能なバリア型同期のためのスケジューリングアルゴリズムとその性能，電子情報通信学会研究会資料，CPSY91-15, pp.91–97 (1991).
4) O'Keefe, M.T. and Dietz, H.G.: Hardware Barrier Synchronization: Static Barrier MIMD (SBM), *1990 Int. Conf. on Parallel Processing*, Vol.1, pp.35–42 (1990).
5) O'Keefe, M.T. and Dietz, H.G.: Hardware Barrier Synchronization: Dynamic Barrier MIMD (DBM), *1990 Int. Conf. on Parallel Processing*, Vol.1, pp.43–46 (1990).
6) 岩根ほか：細粒度マルチプロセッサ MSBM，情報処理学会論文誌，Vol.37, No.6, pp.1196–1205 (1996).
7) 早川ほか：RBCQ 同期機構およびその同期方式の提案と性能評価，情報処理学会論文誌，Vol.39, No.6, pp.1655–1662 (1998).
8) 山家ほか：バリア同期モデル—Taxonomy と新モデルの提案，および，モデル間性能比較，並列処理シンポジウム JSPP'93, pp.119–126 (May. 1993).
9) 岩根ほか：マルチマイクロプロセッサ MDBM/FMM の開発，九州工業大学研究報告（工学），No.68, pp.49–56 (Mar. 1996).
10) 立川ほか：動的バリア同期管理機構をもった並

列計算機 MDBM/FMM，情報処理学会研究報告，96-ARC-119, pp.185–190 (Aug. 1996).
11) 東ほか：バス結合共有メモリ型マルチプロセッサにおける面状バリア同期機構，情報処理学会論文誌，Vol.38, No.11, pp.2398–2401 (1997).
12) 本石：細粒度並列計算機 MSBM とその評価，修士論文，九州工業大学 (1994).
13) 伊理ほか：入門 現代の数学 [13] 計算の効率化とその限界，日本評論社 (1980).
14) 新濃，船田：数値解析の基礎 理論と PAD・PASCAL・C，培風館 (1991).
15) 杉山：多変量データ解析入門，pp.75–79, 朝倉書店 (1983).

(平成 13 年 3 月 16 日受付)
(平成 14 年 2 月 13 日採録)

茶屋道宏貴（正会員）

1972 年生．1992 年鹿児島水産高校専攻科修了．同年九州工業大学工学部電気工学科技官．2000 年九州工業大学工学部電気工学科卒業．計算機アーキテクチャの研究に従事．

岩根　雅彦（正会員）

1946 年生．1968 年京都大学工学部理数工学科卒業．同年（株）東芝に入社．1988 年より九州工業大学教授．工学博士．計算機アーキテクチャの研究に従事．電子情報通信学会会員．IEEE Computer Society 会員．